# Ruby - Feature #19998

## Emit deprecation warnings when the old (non-Typed) Data_XXX API is used

11/10/2023 12:14 PM - byroot (Jean Boussier)

| | | |
|---|---|---|
| **Status:** | Closed | |
| **Priority:** | Normal | |
| **Assignee:** | | |
| **Target version:** | | |

**Description**

The replacement API was introduced in Ruby 1.9.2 (2010) [Feature #3064], and the old untyped data API was marked a deprecated in the documentation as of Ruby 2.3.0 (2015): https://github.com/ruby/ruby/commit/98544c372d948717de22afc86d162e411f1fb5f1

However, as of today, no deprecation warning of any sort is emitted when the API is used.

Ultimately removing the old untyped API would allow to easily reclaim 8B on every T_DATA which could be used by the newly introduced embedded TypedData objects https://github.com/ruby/ruby/pull/7440

Was there any discussions about removing the old API at some point?

Could we emit a warning when the API is used? Either verbose or non-verbose, to at least surface the issue to gem owners? This would help drive the effort to fix outdated gems, allowing to eventually remove this old API in the future.

Looking at our big applications dependencies, there seem to be a small minority of gems still using the old API:

- https://rubygems.org/gems/unf_ext (https://github.com/knu/ruby-unf_ext/pull/72)
- https://rubygems.org/gems/rotoscope (https://github.com/Shopify/rotoscope/pull/99)
- https://rubygems.org/gems/re2 (https://github.com/mudge/re2/pull/116)
- https://rubygems.org/gems/mysql2 (use either API to support old rubies)

Migrating them over doesn't seem too hard.

### Associated revisions

**Revision fbb61a26e71be9faccb4ad2392e71d0a561854d1 - 06/06/2024 09:44 AM - byroot (Jean Boussier)**

Mark old Data API as deprecated

[Feature #19998]

**Revision fbb61a26e71be9faccb4ad2392e71d0a561854d1 - 06/06/2024 09:44 AM - byroot (Jean Boussier)**

Mark old Data API as deprecated

[Feature #19998]

**Revision fbb61a26 - 06/06/2024 09:44 AM - byroot (Jean Boussier)**

Mark old Data API as deprecated

[Feature #19998]

**Revision ec7babd12d5802e9970d9aecd78ac2e5ec79bcc8 - 06/06/2024 03:49 PM - nobu (Nobuyoshi Nakada)**

[Feature #19998] Untyped Data API has been marked as deprecated

**Revision ec7babd12d5802e9970d9aecd78ac2e5ec79bcc8 - 06/06/2024 03:49 PM - nobu (Nobuyoshi Nakada)**

[Feature #19998] Untyped Data API has been marked as deprecated

**Revision ec7babd1 - 06/06/2024 03:49 PM - nobu (Nobuyoshi Nakada)**

[Feature #19998] Untyped Data API has been marked as deprecated

### History

**#1 - 11/10/2023 12:25 PM - byroot (Jean Boussier)**

*- Description updated*

#### #2 - 11/10/2023 01:07 PM - Eregon (Benoit Daloze)

Maybe we should deprecate the corresponding C functions instead/in addition to runtime warnings?
i.e. the fix is in C extensions, not Ruby code.

+1 on deprecating those, it would simplify to have one (TypedData) instead of two data-like types.

#### #3 - 11/10/2023 02:14 PM - byroot (Jean Boussier)

*- Description updated*

#### #4 - 11/10/2023 02:16 PM - byroot (Jean Boussier)

> Maybe we should deprecate the corresponding C functions instead/in addition to runtime warnings?

Yes, I don't have a string opinion on the specific. I'm happy to implement whatever Matz decide.

#### #5 - 11/10/2023 11:13 PM - rubyFeedback (robert heiler)

I do not have anything against the proposal itself, but in
regards to "This would help drive the effort to fix outdated
gems", I believe this to be slightly too optimistic. Many gems
that have not been updated in, say, 2-3 years, are probably
no longer (actively) maintained (for the most part). One can
see that on gems hosted on rubygems.org: many gems have one
peak activity of several releases, and then no more update
for years to come. So I believe the primary (better) argument
to be made should be rather in regards to "reclaim 8B on every
T_DATA" or deprecation handling in general (e. g. since as of
the year 2010) than assume adoption of older gems. If the ruby
devs communicate a strategy of change - and have given enough
time to adjust - then I think ruby should move forward towards
change (in general, that is; again, I don't have any particular
opinion on this issue; I don't think it affects me either way,
though of course reading "reclaiming xyz" sounds great - as matz
once said, nobody minds more speed/efficiency).

#### #6 - 11/11/2023 11:50 PM - byroot (Jean Boussier)

*- Description updated*

#### #7 - 11/13/2023 09:04 AM - byroot (Jean Boussier)

Interesting tidbits from the re2 PR: https://github.com/mudge/re2/pull/116#pullrequestreview-1725029520

> I was unaware of the newer TypedData API.

Which IMO confirms that just marking something as deprecated in the documentation has little effect, and some form of deprecation warning would help at least some gems upgrade.

#### #8 - 11/13/2023 12:53 PM - byroot (Jean Boussier)

*- Description updated*

#### #9 - 12/07/2023 04:48 AM - naruse (Yui NARUSE)

Generally deprecation and migration takes development cost for gem developer.
Therefore such proposal needs to clearly shows the merit of the removal.
("marked as deprecated" is not such reason)

About the strategy of C API deprecation, runtime warning is not a good way because such warning make the gem unusable.
As byroot originally says first it should show a build time warning, then just remove the API is reasonable.
If the migration really has the benefit, gem owners should fix the usage.

#### #10 - 12/07/2023 07:13 AM - byroot (Jean Boussier)

> Therefore such proposal needs to clearly shows the merit of the removal.

I mentioned why I'd want it removed. It would allow to reclaim 8B per objects, allowing T_DATA to be 32B rather than 40B, and we have plans to make slot sizes powers of two.

#### #11 - 12/07/2023 07:14 AM - byroot (Jean Boussier)

But a build time warning at least would be a good way to prevent new gems from being written using that API.

#### #12 - 12/20/2023 04:56 AM - ko1 (Koichi Sasada)

I agree to show build-time warning but I don't think we can remove T_DATA feature.

#### #13 - 12/20/2023 08:23 AM - byroot (Jean Boussier)

Alright. A big part of the API is composed of macros, so I'm not too sure how we can effectively make them emit a build time warning, but I'll see what I can do.

#### #14 - 06/06/2024 09:35 AM - mame (Yusuke Endoh)

Discussed at the dev meeting. Matz accepted build-time warning.

However, the author of a gem that is not well maintained is not likely to notice the build-time warning. Whether it is actually possible to remove the APIs may require careful consideration in the future.

#### #15 - 06/06/2024 09:42 AM - byroot (Jean Boussier)

> Whether it is actually possible to remove the APIs may require careful consideration in the future.

Agreed. I'm not suggesting that yet. However I do have a proof of concept that degrades a bit the old API performance in favor of improving the new API performance, so having the API explicitly marked as deprecated help justify this kind of changes.

#### #16 - 06/06/2024 09:44 AM - byroot (Jean Boussier)

*- Status changed from Open to Closed*

Applied in changeset git|fbb61a26e71be9faccb4ad2392e71d0a561854d1.

---

Mark old Data API as deprecated

[Feature #19998]